Energy Solutions Germany

Deep learning workshop

In [94]:
```
print("hello")
```

```
hello
```

Welcome to the interactive ML workshop!

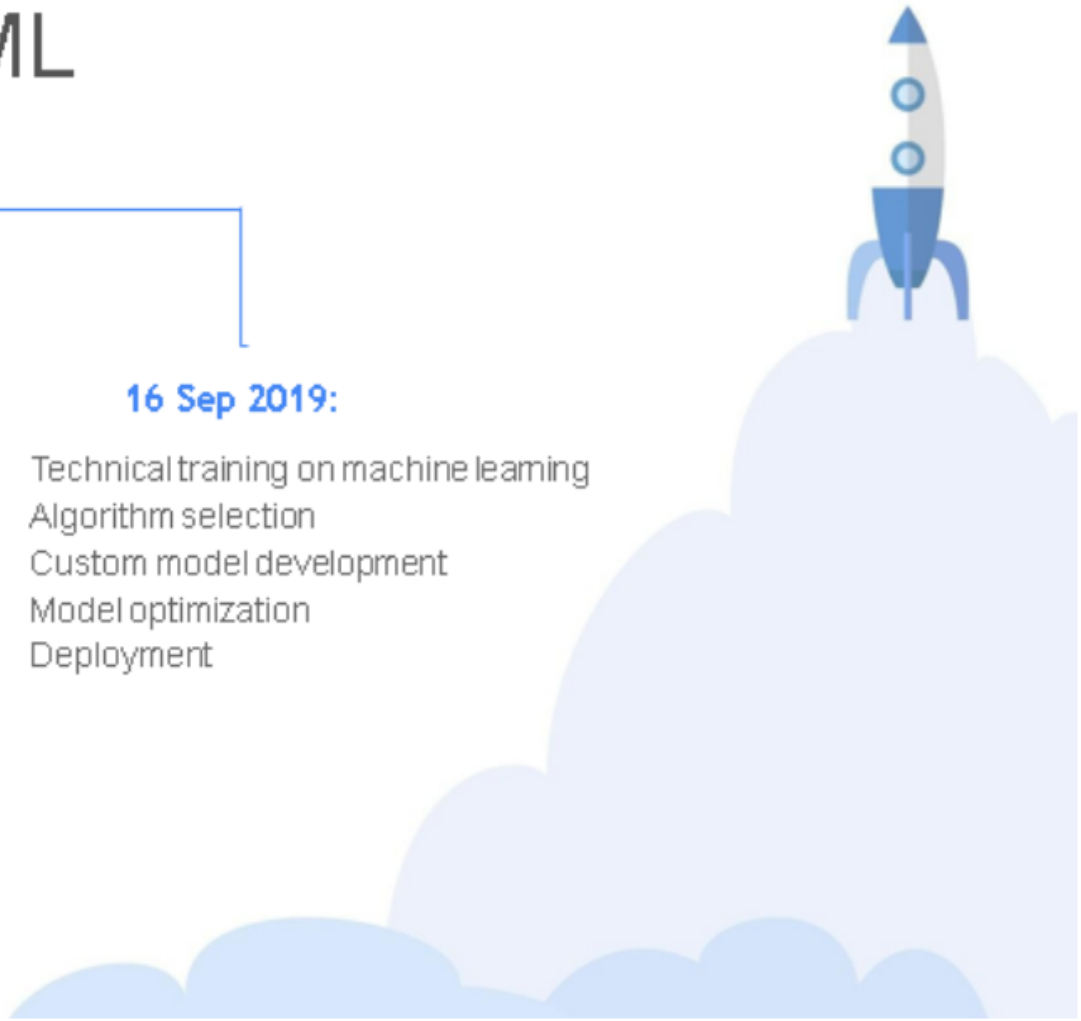
Discover ML

4 Sep 2019:

- Innovation
- Ideation
- ML use case discovery
- Data exploration
- Use case prioritization

16 Sep 2019:

- Technical training on machine learning
- Algorithm selection
- Custom model development
- Model optimization
- Deployment



Workshop Agenda

1. Welcome
Define goals of workshop, meet each other

2. Objectives
Come to a common understanding of what ML is and what we want to achieve with it

3. Raspberry Pi
Demo of a neural network in a box – facial recognition and style transfer

4. ML Concepts
Basic intro to important ML steps

5. On-screen training
Building a classifier with computer vision

6. Hands-on replication
Do the same but on a different target

7. Conclusion and progress update
Where are we on the gas imbalance front and what have we learned?

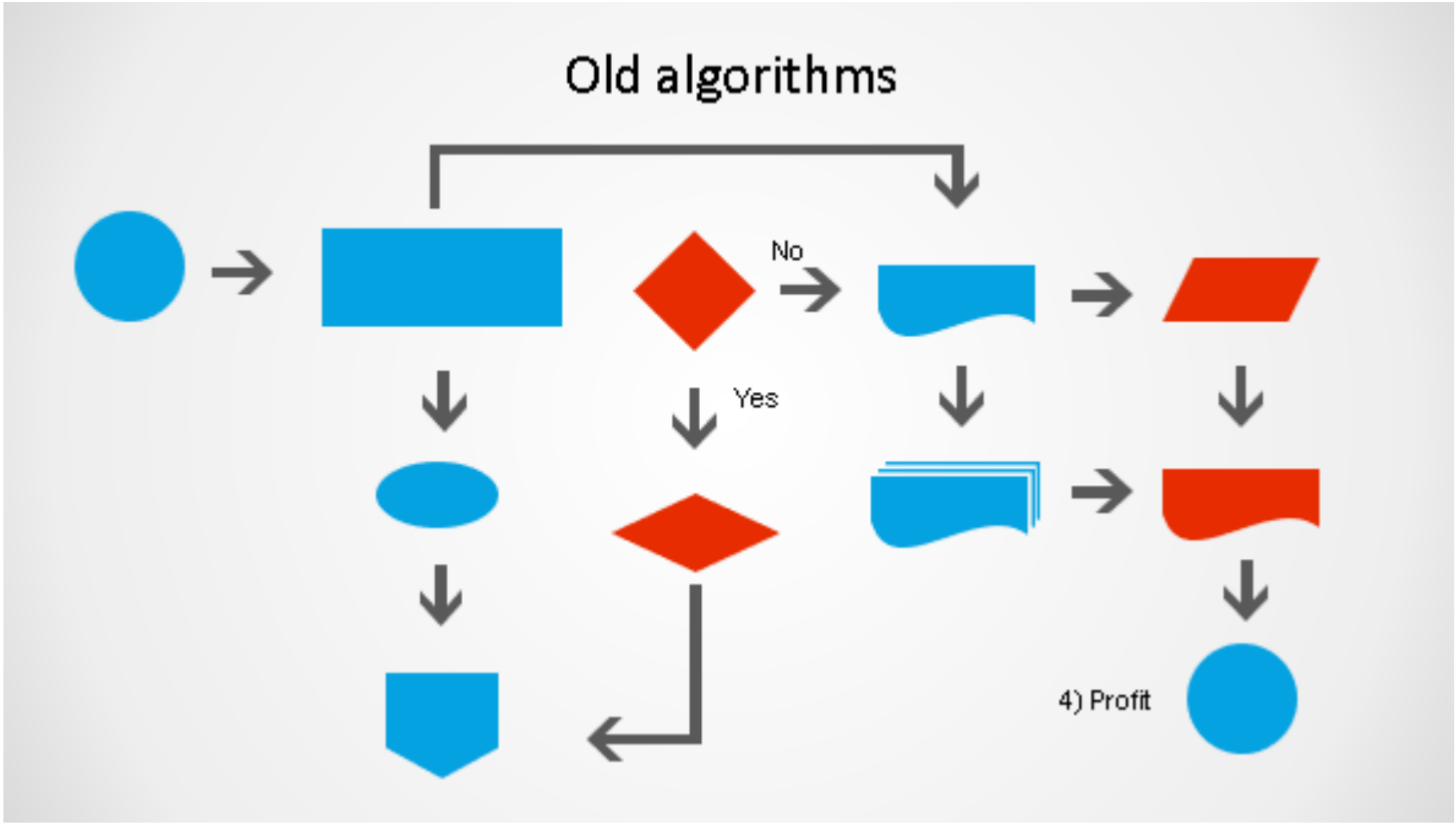
We are using a jupyter notebook, which is a standard interactive, collaborative data science toolkit

The name refers to Julia, Python, and R, which are popular data science languages

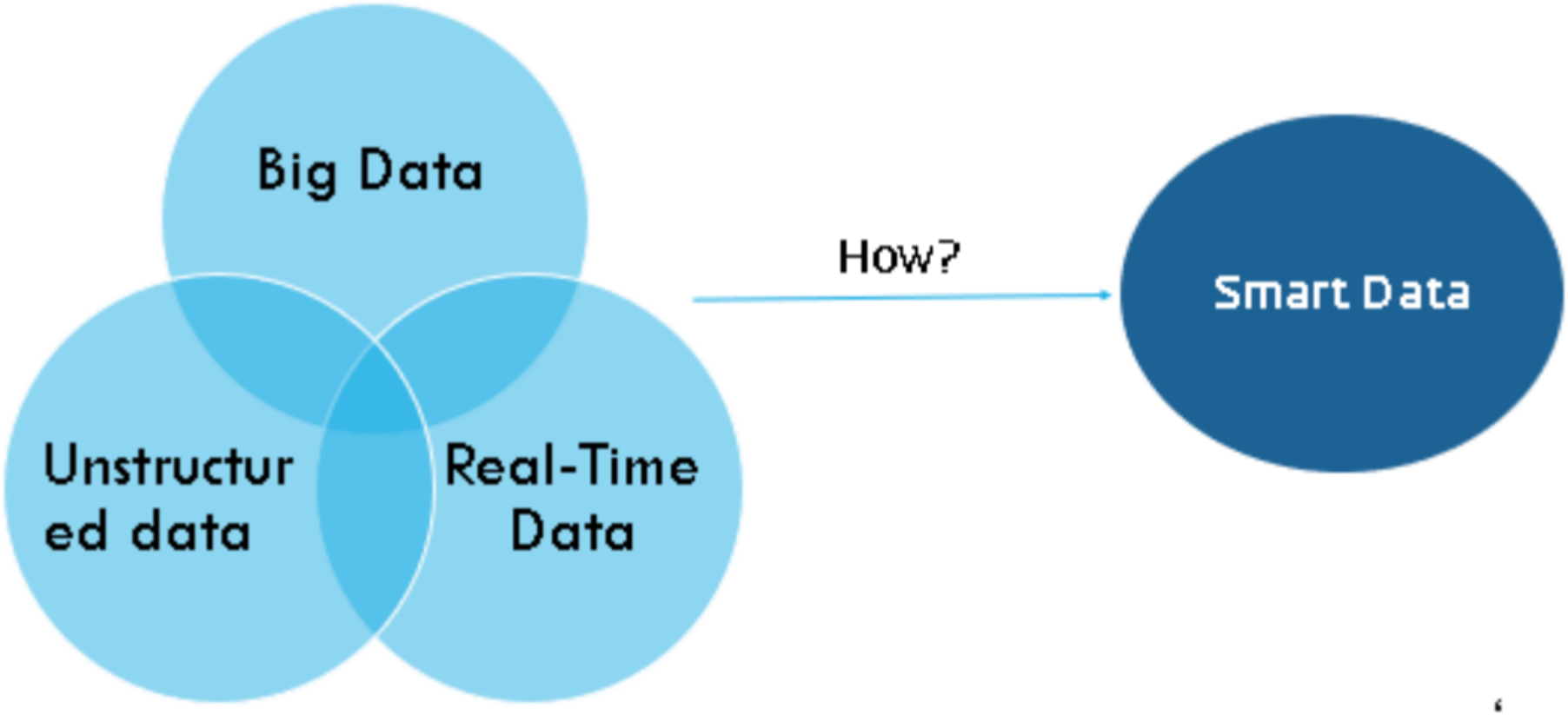
In this workshop we will use the python language, which was developed by Dutchie Guido van Rossum.

It is the most popular, and fastest growing data science language.

So why do we need machine learning?



We are drowning in data, neither humans nor "old" algorithms are able to deal with big data



The popular imagination of what ML is

Lots of data

Complex mathematics in multidimensional spaces

Magical results

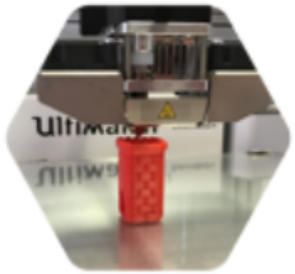
What you will find after this workshop is that ML is….

Define objectives

Collect data

Understand and prepare the data

Create and evaluate the model

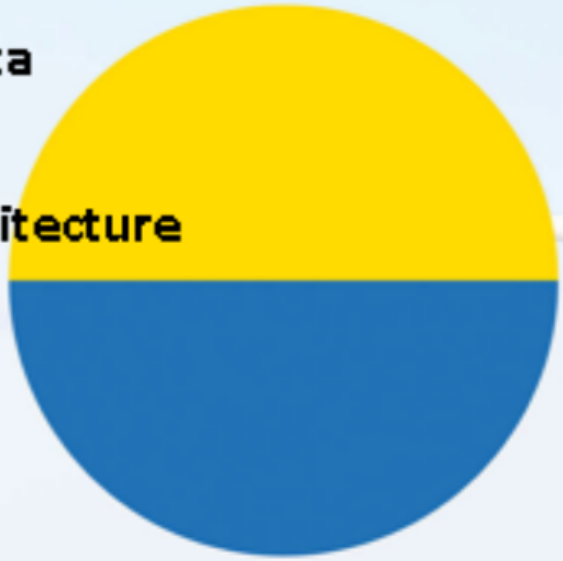


Refine the model

Serve and monitor the model

ML model building steps

1. Gather relevant data
2. Data preparation
3. Choose model architecture
4. Train the model
5. Evaluate the model
6. Tune the model
7. Predict!



How can we get answers from data?

- Model = a system that answers questions
- Model is built by “training”
- Goal of training is to build a model that is as accurate as possible
- Data required to be accurate depends on the question

Increasing depth in data

- **Dummy = Yes or No, 0 or 1, True or False**
- **Scalar = A number, for example 255 kWh**
- **Vector = A time series of kWh**
- **Matrix = A vector of kWh by location**
- **Tensor = A matrix multiplied by at least on matrix (Multi-dimensional array), for example ESG's Loctions list as XY coordinates attachted to the time-series matrix**

Key takeaways

1. **Up to date about imbalance progress**
2. **Understand AI key concepts - transfer learning, model architecture, data requirements**
3. **Able to do some AI yourself**

In this workshop we will do "computer vision", one particular area of machine learning

This is good to work with interactively because you can see with your own eyes what is going on in the neural network...

... all concepts apply to other areas of machine learning, and artificial intelligence more broadly

From vision, to speech, text, and numbers

In [1]:
```
#Let's start by setting up the environment that we will use during this workshop

import os, requests, json, os.path
from fastai.vision import *
from fastai import *
from fastai.callbacks.hooks import *
#from google_images_download import google_images_download
#response = google_images_download.googleimagesdownload()
from PIL import Image as pil_image
PIL.Image.LOADTRUNCATEDIMAGES = True
```

Ok, so what do we want to classify on?

In [95]:
```
os.chdir('C:/images/')
os.makedirs('classes', exist_ok=True)
path = 'C:/images/classes/'
class_list = [(f,f.replace('.csv','')) for f in os.listdir(path) if '.csv' in f]
for _,c in class_list:
    loc = os.path.join(path,c)
    if not os.path.isdir(loc):
        os.mkdir(loc)
print(class_list)
```

```
[('chp mtu.csv', 'chp mtu'), ('chp mwn.csv', 'chp mwn'), ('jenbacher_chp.csv', 'jenbacher_chp')]
```

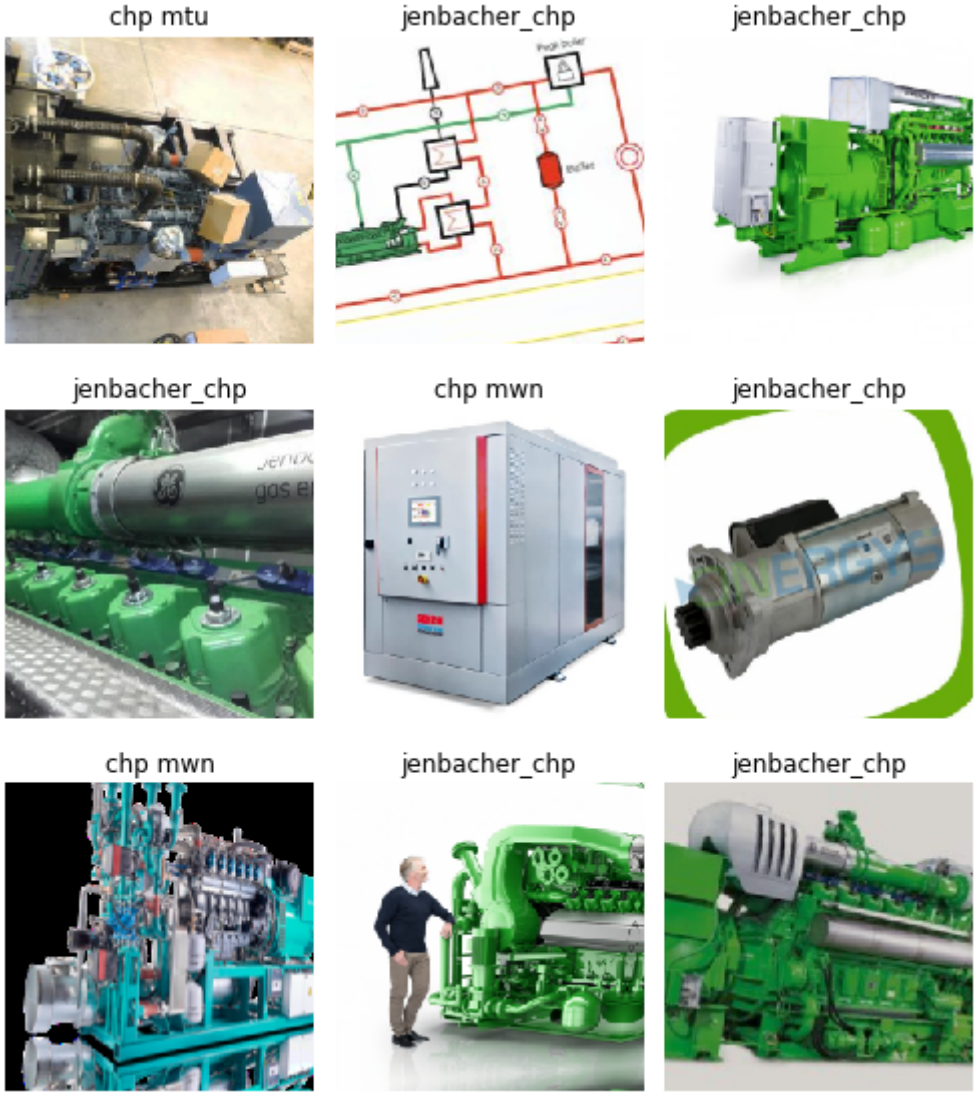
In [96]:
```
for f,c in class_list:
    print(f,c)
    csv = os.path.join(path,f)
    print(csv)
    download_images(csv,os.path.join(path,c))
    verify_images(os.path.join(path,c), delete=True, max_size=500)
```

In [97]:
```
# do we have enough data?

DIR = 'C:/images/classes/'
d ={name:len(os.listdir(os.path.join(DIR,name))) for name in os.listdir(DIR) if not os.path.isfile(os.pat
h.join(DIR, name))}
for k,v in d.items():
    print(k,':',v)
    print('')
```

```
chp mtu : 62

chp mwn : 88

jenbacher_chp : 91
```

In [98]:
```
# prepping the data...

np.random.seed(42)
data = ImageDataBunch.from_folder("C:/images/", train=".", valid_pct=0.2,
        ds_tfms=get_transforms(), size=224, num_workers=4).normalize(imagenet_stats)
```

In [99]:
```
# is everything ok?

data.show_batch(rows=3, figsize=(7,8))
```



In [100]:
```
len(data.train_ds), len(data.valid_ds)
```

Out[100]:
```
(190, 47)
```

In [107]:
```
# so here we choose the model architecture!

learn = cnn_learner(data, models.resnet34, metrics=[error_rate, accuracy])
```

In [ ]:
```
learn.load('pre_trained_chp')
```

In [109]:
```
learn.fit_one_cycle(2)
```

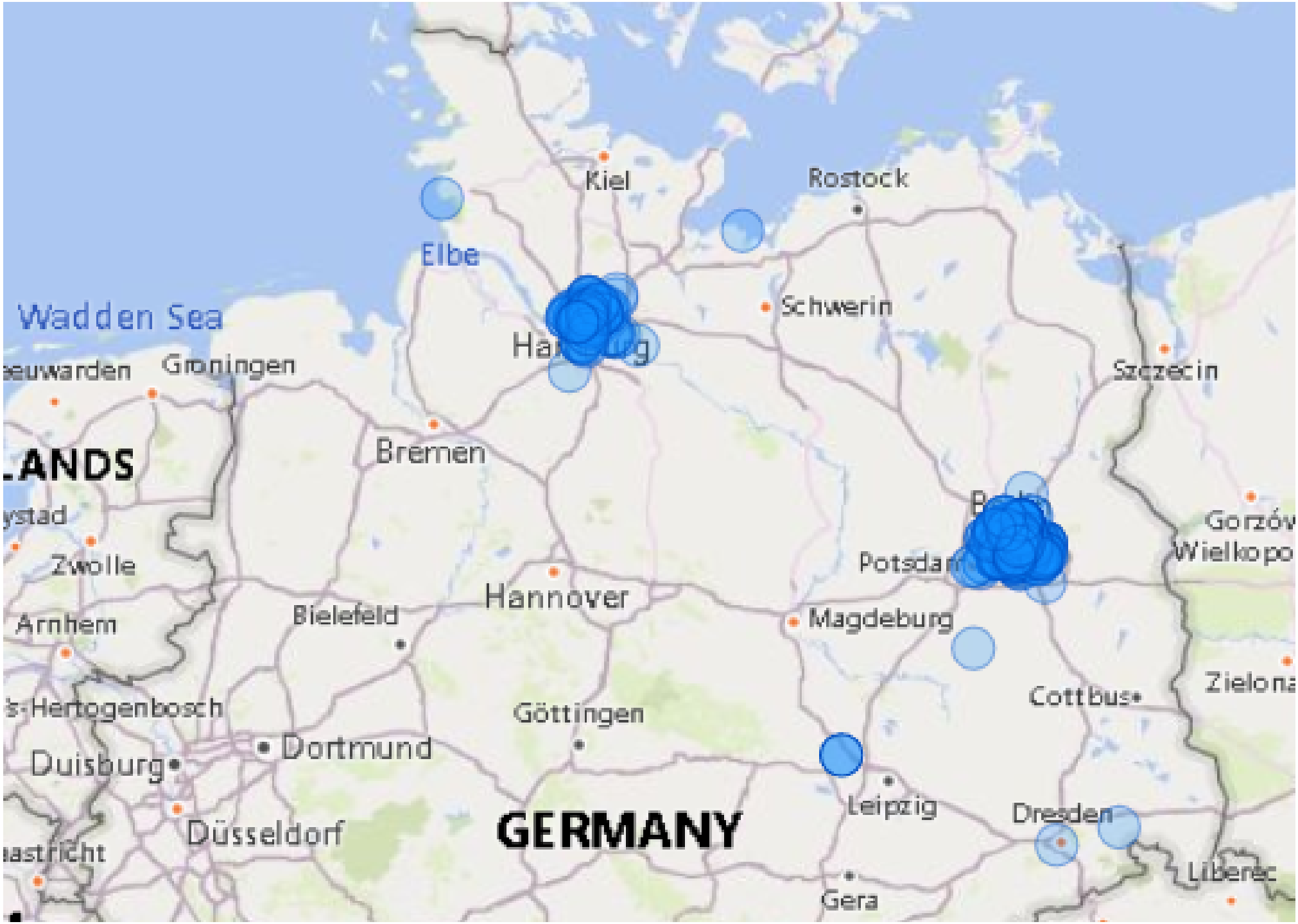
epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.633854	1.757635	0.340426	0.659574	01:48
1	0.451185	1.498469	0.297872	0.702128	01:50

In [110]:
```
learn.validate()
```

Out[110]:
```
[1.4984691, tensor(0.2979), tensor(0.7021)]
```

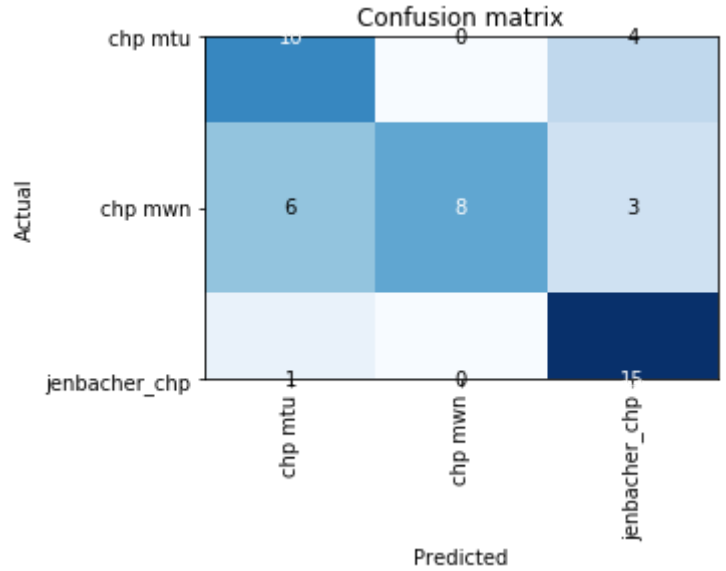

Kiel
Rostock
Elbe
Wadden Sea
Schwerin
Groningen
Szczecin
Bremen
Zwolle
Potsdam
Hannover
Arnhem
Bielefeld
Magdeburg
Cottbus
's-Hertogenbosch
Göttingen
Duisburg
Dortmund
Leipzig
Dresden
Düsseldorf
GERMANY
Gera
Ústí nad Labem
Wiesbaden
Frankfurt
Carlsbad
Mainz
Prague
Pilsen
Nuremberg
Regensburg
Budejovice
Saarbrücken
Karlsruhe
Nancy
Stuttgart
Strasbourg
Munich
Linz
Basel
Schaffhausen
Salzburg
Zürich
Sankt Gallen
Innsbruck
AUSTRIA
Neuchâtel
Lucerne
LIECHTENSTEIN
SWITZERLAND

While we're waiting - what's actually going on in this neural network?


Kiel
Rostock
Elbe
Wadden Sea
Schwerin
Szczecin
Groningen
Bremen
Zwolle
Hannover
Arnhem
Bielefeld
Magdeburg
Potsdam
Cottbus
Göttingen
Dortmund
Duisburg
Leipzig
Dresden
Düsseldorf
GERMANY
Gera
Liberec


Ústí nad Labem
Carlsbad
Prague
Pilsen
CZECH RE
Wiesbaden
Frankfurt
Mainz
LUXEMBOURG
Nuremberg
Regensburg
Saarbrücken
Karlsruhe
Budejovice
Stuttgart
Nancy
Strasbourg
Linz
Sankt
Munich

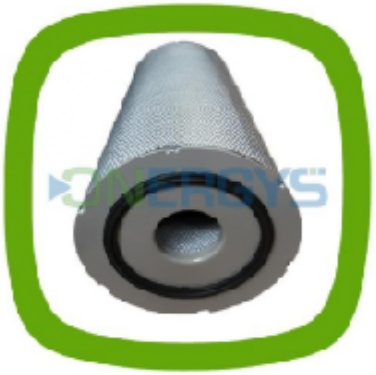
In [111]:
```
interp = ClassificationInterpretation.from_learner(learn)
interp.plot_confusion_matrix()
```



In [112]:
```
interp.plot_top_losses(9)
```

prediction/actual/loss/probability

jenbacher_chp/chp mtu / 9.08 / 0.00



chp mtu/chp mwn / 8.00 / 0.00

jenbacher_chp/chp mwn / 7.69 / 0.00

chp mtu/chp mwn / 7.49 / 0.00

chp mtu/chp mwn / 6.17 / 0.00

jenbacher_chp/chp mtu / 5.18 / 0.01

jenbacher_chp/chp mwn / 4.78 / 0.01

jenbacher_chp/chp mwn / 4.77 / 0.01

chp mtu/chp mwn / 4.42 / 0.01



In [105]:
```
# so what would happen with random weights, instead of starting with a pre-trained network (transfer lear
ning)?

learn.save('pre_trained_chp')
del learn
```

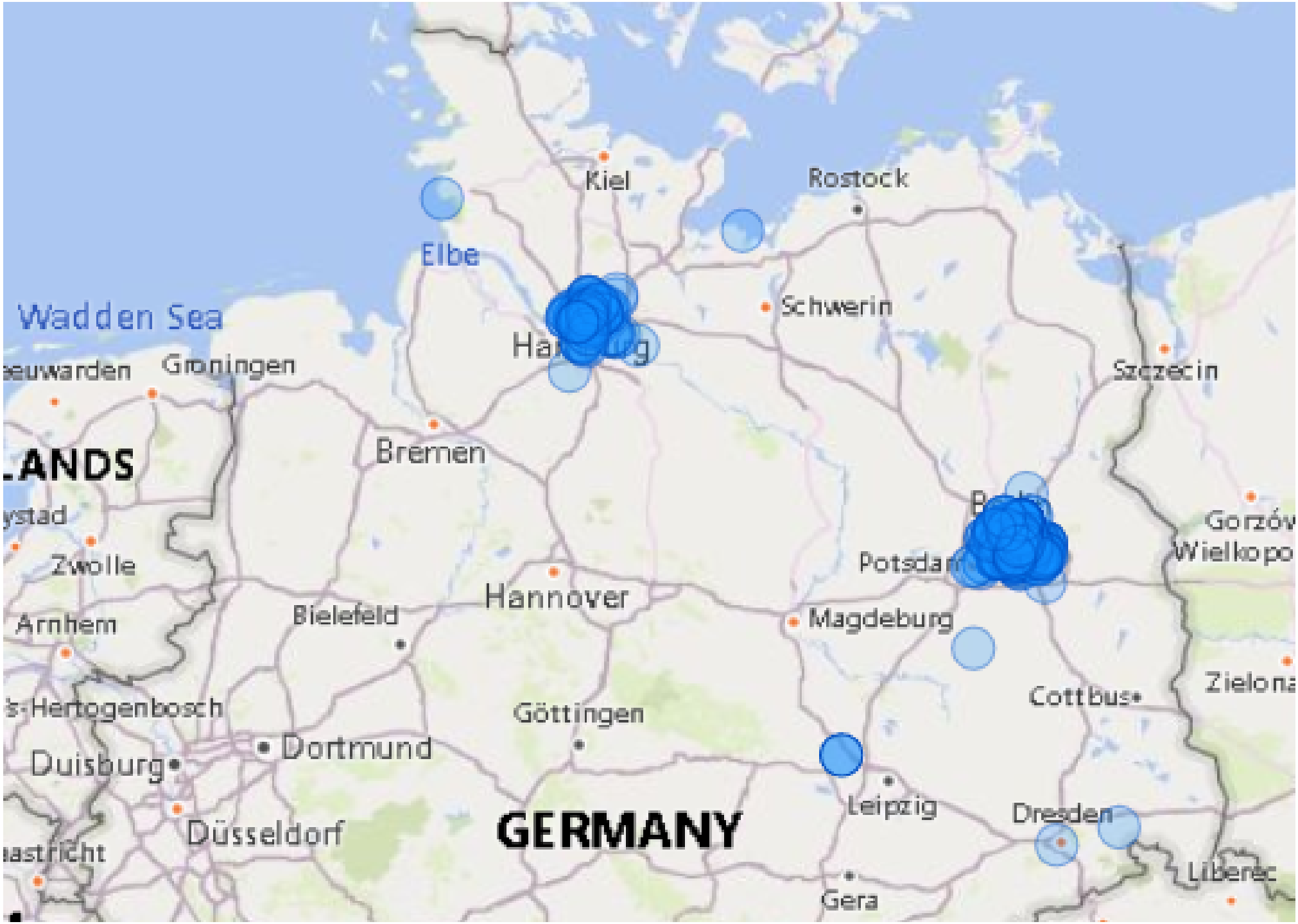
In [81]:
```
learn_wo = cnn_learner(data, models.resnet34, metrics=[error_rate,accuracy], pretrained=False)
```

```
c:\program files\python37\lib\site-packages\fastai\vision\learner.py:106: UserWarning: `create_cnn` is dep
recated and is now named `cnn_learner`.
  warn("`create_cnn` is deprecated and is now named `cnn_learner`.")
```

In [82]:
```
learn_wo.fit_one_cycle(4)
```

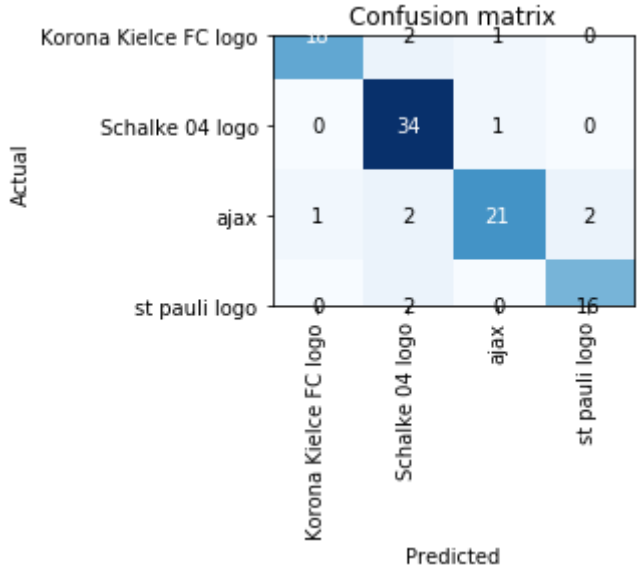
epoch	train_loss	valid_loss	error_rate	accuracy	time
0	2.383491	6.885210	0.720000	0.280000	05:09
1	1.988807	1.177455	0.480000	0.520000	05:05
2	1.761498	1.375785	0.520000	0.480000	05:05
3	1.619428	1.087470	0.430000	0.570000	05:05

What is fitting and why does it matter?

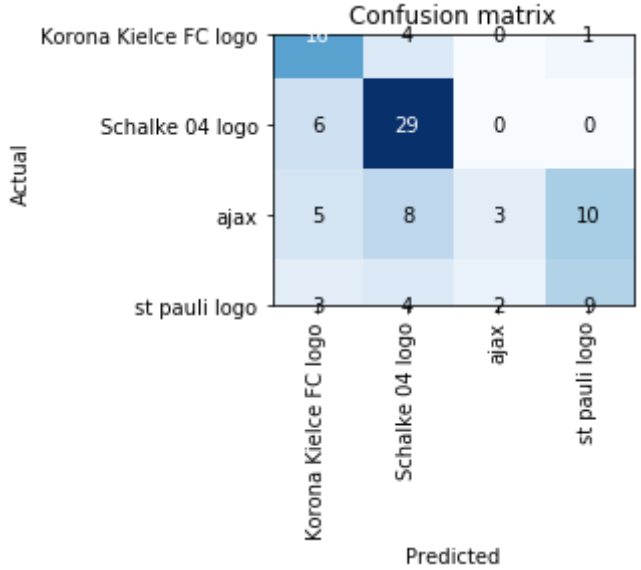

Kiel
Rostock
Elbe
Wadden Sea
Schwerin
Szczecin
Groningen
Bremen
Zwolle
Hannover
Arnhem
Bielefeld
Magdeburg
Potsdam
Cottbus
Göttingen
Dortmund
Duisburg
Leipzig
Dresden
Düsseldorf
GERMANY
Gera
Liberec

In [83]:
```
interp_wo = ClassificationInterpretation.from_learner(learn_wo)
```

In [84]:
```
# not so good in comparison

interp.plot_confusion_matrix()
plt.figure()
interp_wo.plot_confusion_matrix()
```



<Figure size 432x288 with 0 Axes>



In [85]:
```
interp_wo.plot_top_losses(9)
```

prediction/actual/loss/probability

Korona Kielce FC logo/ajax / 4.31 / 0.01

Schalke 04 logo/ajax / 4.17 / 0.02

Korona Kielce FC logo/Schalke 04 logo / 3.48 / 0.03

Schalke 04 logo/Korona Kielce FC logo / 3.07 / 0.05

Schalke 04 logo/ajax / 3.07 / 0.05

Schalke 04 logo/st pauli logo / 2.99 / 0.05

Korona Kielce FC logo/ajax / 2.78 / 0.06

Schalke 04 logo/st pauli logo / 2.74 / 0.06

Korona Kielce FC logo/Schalke 04 logo / 2.71 / 0.07



In [86]:
```
learn_wo.save('trained')
```

In [ ]:
```
learn.load('pre_trained')
```

In [87]:
```
m = learn_wo.model.eval();
```

In [88]:
```
idx=18
x,y = data.valid_ds[idx]
x.show()
x,_ = data.one_item(x)
```

In [89]:
```
x_im = Image(data.denorm(x)[0])
#Image(data.denorm(x)[0])
```

In [90]:
```
def hooked_backward(cat=y):
    with hook_output(m[0]) as hook_a:
        with hook_output(m[0], grad=True) as hook_g:
            preds = m(x)
            preds[0,int(cat)].backward()
    return hook_a,hook_g

hook_a,hook_g = hooked_backward()
```

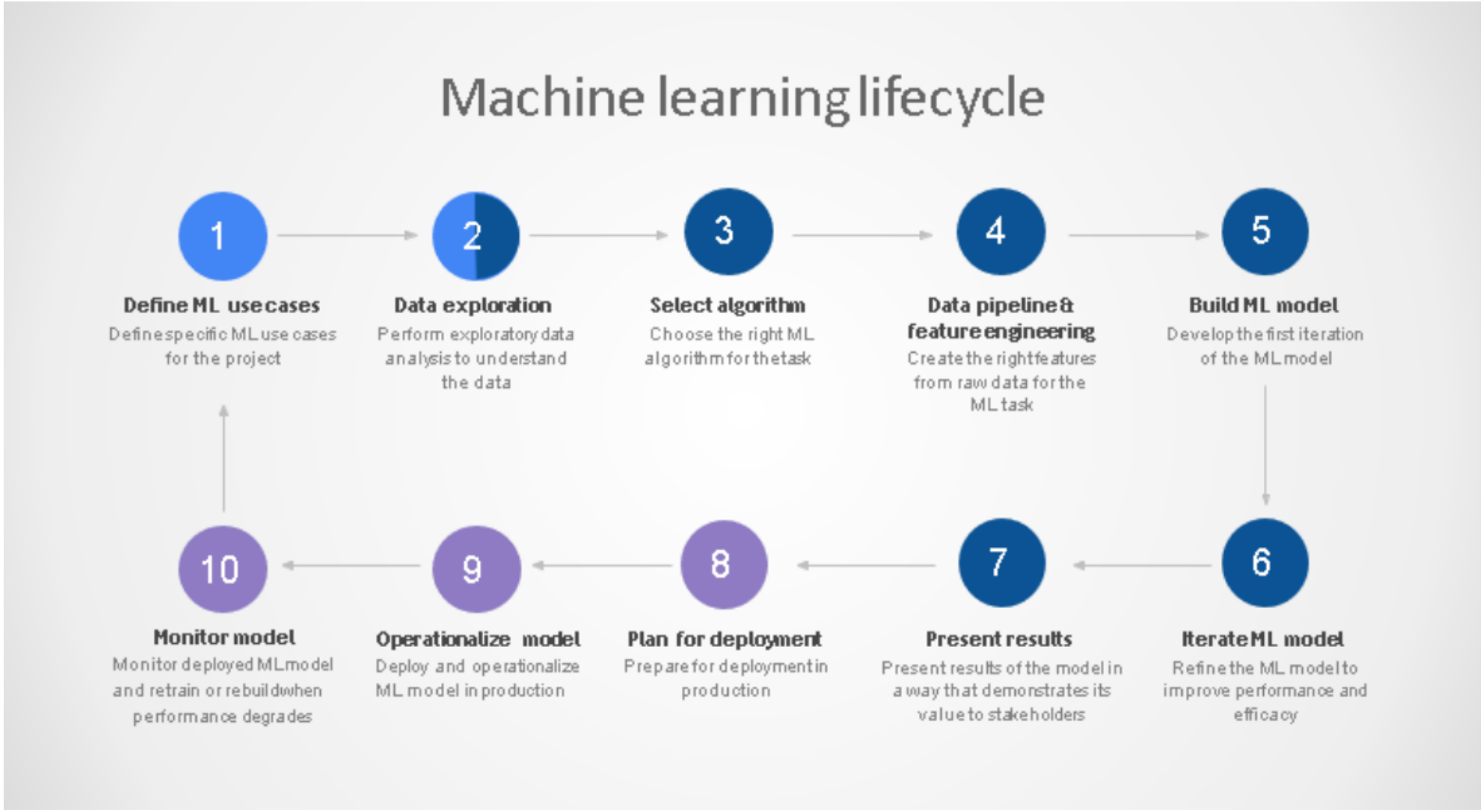
In [91]:
```
acts  = hook_a.stored[0].cpu()
acts.shape
avg_acts = acts.mean(0)
avg_acts.shape
```

Out[91]:
```
torch.Size([7, 7])
```

In [92]:
```
def show_heatmap(hm):
    _,ax = plt.subplots()
    x_im.show(ax)
    ax.imshow(hm, alpha=0.6, extent=(0,224,224,0),
              interpolation='bilinear', cmap='magma');
```

In [93]:
```
show_heatmap(avg_acts)
```


S
04


Machine learning lifecycle
1
Define ML use cases
Define specific ML use cases for the project
2
Data exploration
Perform exploratory data analysis to understand the data
3
Select algorithm
Choose the right ML algorithm for the task
4
Data pipeline & feature engineering
Create the right features from raw data for the ML task
5
Build ML model
Develop the first iteration of the ML model
6
Iterate ML model
Refine the ML model to improve performance and efficacy
7
Present results
Present results of the model in a way that demonstrates its value to stakeholders
8
Plan for deployment
Prepare for deployment in production
9
Operationalize model
Deploy and operationalize ML model in production
10
Monitor model
Monitor deployed ML model and retrain or rebuild when performance degrades

Make ML models do the hard work for you

What do successful AI projects have in common?

1. There is a tangible and recognized business problem that can be addressed with data science
2. The problem is urgent and there is willingness to dedicate substantial resources to fix it
3. There is abundant high-quality labelled data
4. There is deep and passionate domain knowledge
5. Incremental-iterative improvements to the model can be made easily
6. The data is on a scalable and stable data platform
7. Results of the model can be measured and applied in the real world

Takeaway concepts

- **Model architecture**
- **Transfer learning**
- **Freezing**

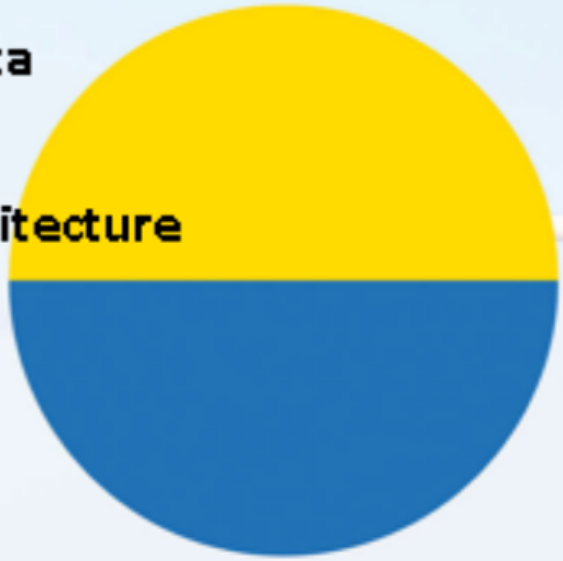
- **Parameters/weights/coefficients**
- **Over/underfitting Fitting**
- **Stochastic Gradient descent**

- **Loss function**
- **Training batches**
- **Learning rate**

Now you! We will supervise

ML model building steps

1. Gather relevant data
2. Data preparation
3. Choose model architecture
4. Train the model
5. Evaluate the model
6. Tune the model
7. Predict!



Machine Learning Use Cases

Manufacturing

- Predictive maintenance or condition monitoring
- Warranty reserve estimation
- Propensity to buy
- Demand forecasting
- Process optimization
- Telematics

Retail

- Predictive inventory planning
- Recommendation engines
- Upsell and cross-channel marketing
- Market segmentation and targeting
- Customer ROI and lifetime value

Healthcare and Life Sciences

- Alerts and diagnostics from real-time patient data
- Disease identification and risk satisfaction
- Patient triage optimization
- Proactive health management
- Healthcare provider sentiment analysis

Travel and Hospitality

- Aircraft scheduling
- Dynamic pricing
- Social media – consumer feedback and interaction analysis
- Customer complaint resolution
- Traffic patterns and congestion management

Financial Services

- Risk analytics and regulation
- Customer Segmentation
- Cross-selling and up-selling
- Sales and marketing campaign management
- Credit worthiness evaluation

Energy, Feedstock and Utilities

- Power usage analytics
- Seismic data processing
- Carbon emissions and trading
- Customer-specific pricing
- Smart grid management
- Energy demand and supply optimization

Thank you!